New Approach for Modifying Blowfish Algorithm by Using Multiple Keys

Afaf M. Ali Al-Neaimi[†], Rehab F. Hassan[††]

[†]Department of Software Engineering, AL-Zaytoonah University, Amman, Jordan
[††]Department of Computer Science, University of Technology, Baghdad, Iraq

Summary
In this paper, cellular automata (CAs) are used to design a symmetric key cryptography system based on Bluefish algorithm, CAs are applied to generate a multiple pseudo-random numbers sequence (PNS) which is used during the encryption process. The quality of PNSs highly depends on the set of applied CA rules. This paper introduces a new method to enhance the performance of the Bluefish Algorithm. This is done by building a new structure for the 16 rounds in the original algorithm by replacing the OR operation with a new introduced operation. This structure makes use of multiple secrete keys. The principle of Cellular Automata (CA) is used to generate these multiple keys in a simple and effective way. The proposed method provides high quality encryption, and the system is very resistant to attempts of breaking the cryptography key.

Key words:
Computer security, Bluefish Algorithm, Cellular Automata, triple data encryption algorithm.

1. Introduction

Pseudo-random number sequences are needed in many important applications, such as Monte Carlo techniques, Brownian dynamics, and stochastic optimization methods. Cellular automata (CA) offer a number of advantages over other methods as random number generators such as algorithmic simplicity and easy hardware implementation.

Cellular automata were first introduced by von Neumann and later by Wolfram [9] as simple models for physical, biological and computational systems. CA have previously been used as encrypting devices by Wolfram and by Nandi, Kar, Gutowitz and Guam [3, 4] used CA for publickey cryptography. The pseudo-random sequence generator based on CA has been extensively studied in the last decades. In 1986, Wolfram [9] first applied CA in pseudorandom number generation. After, other authors as Hortensius, Tsalides, Sipper and Perrenoud used the CA to generate the pseudorandom sequences used in cryptography [2, 8]. But those methods are based on artificial methods to construct CA rules. The main disadvantage of those methods is that they are involved in large tedious work. Another valid method is introduced by Sipper and Tomassini [8], they used genetic algorithm to find the best CA randomizer rules automatically. A series of research work has been done to generate pseudo-random sequences. The rest of the paper is organized as follows. In Section 2 we introduce some basic concepts about cellular automata; Section 3 gives a full description with the algorithm of the DES algorithm. In section 4, the method of generating the multiple random keys is explained; the rest sections explain the proposed improvement to the DES algorithm with the implementation and the conclusion of this improvement.

The search for new public-key schemes, improvements to existing cryptographic mechanisms, and proofs of security continues at a rapid pace. Various standards and infrastructures involving cryptography are being put in place. Security products are being developed to address the security needs of an information intensive society [9]. The research is an attempt to improve most cryptographic algorithms that depend on using logical OR operation, then the DES algorithm is explained briefly in section 3, where the proposed improvement will be applied on, section 4 contains a full description to the operation that will be exchanged with the ordinary OR operation. The rest of the paper explains the new improved DES algorithm, with the conclusion and the suggested future work that can be applied.

2. Elementary Cellular Automata

A grid of elementary cellular automata consists of a one-dimensional row of cells, where each cell can be in one of two states, and the rules for the transformation of a cell are based on the current state of the cell and its two closest neighbors. The neighborhood thus consists of three cells. [2, 8]
The following is an example of a set of rules for elementary cellular automata where the two states are called "white" and "black:"

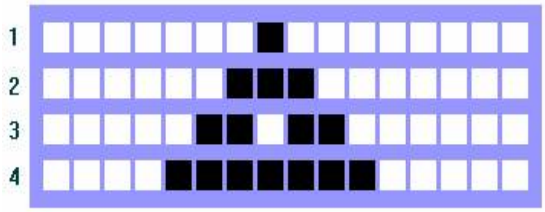
1. If all three cells are white, the cell remains white.

2. If all three cells are black, the cell becomes white.

Manuscript received March 5, 2011
Manuscript revised March 20, 2011

3. In any other case (that is, if there is a mixture of black and white cells in the neighborhood), the cell becomes (or remains) black.

The following diagram shows four successive generations of cellular automata conforming to these rules—an initial generation consisting of one black cell with all other cells white, and three subsequent generations created by three applications of the rules:



This example illustrates three basic properties of cellular automata described by Rennard [4]:

- Parallelism: The cells all change state independently and simultaneously (that is, in parallel).
- Locality: The new state of a cell is dependent upon only the state of the cell itself and the states of its neighbors.
- Homogeneity: All cells have an identical set of possible states and follow the same rules.

3. Blowfish Algorithm

Blowfish is a symmetric block cipher that can be used as a drop-in replacement for DES or IDEA. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for both domestic and exportable use. Blowfish was designed in 1993 by Bruce Schneier as a fast, free alternative to existing encryption algorithms. Since then it has been analyzed considerably, and it is slowly gaining acceptance as a strong encryption algorithm [4].

Blowfish is a variable-length key, 64-bit block cipher. The algorithm consists of two

Parts: a key-expansion part and a data- encryption part. Key expansion converts a key of at most 448 bits into several subkey arrays totaling 4168 bytes. Data encryption occurs via a 16-round Feistel network. Each round consists of a key dependent Permutation, and a key- and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookups per round.

Diagram shown in Figure (1) shows the action of Blowfish. Where Blowfish has 16 rounds, The input is a 64-bit data element,

x. Divide x into two 32-bit halves: xL, xR.

Then, for i = 1 to 16:

xL = xL XOR Pi

xR = F(xL) XOR xR

Swap xL and xR

After the sixteenth round, swap xL and xR again to undo the last swap.

Then, xR = xR XOR P17 and xL = xL XOR P18.

Finally, recombine xL and xR to get the ciphertext [4, 5].

F-function splits the 32-bit input into four eight-bit quarters, and uses the quarters as input to the S-boxes. The outputs are added modulo 232 and XORed to produce the final 32-bit output. Since Blowfish is a Feistel network, it can be inverted simply by XORing P17 and P18 to the ciphertext block, then using the P-entries in reverse order.

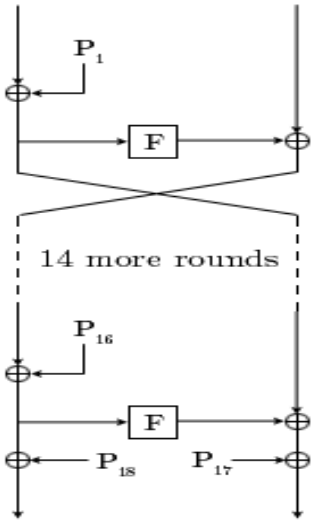


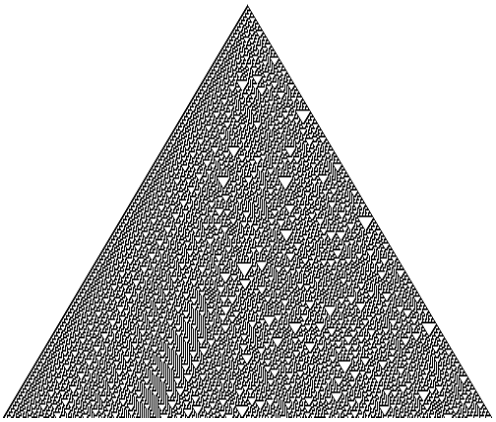
Figure (1): Blowfish each round action

4. Random keys generated by CA

The simplest nontrivial CA would be one-dimensional, with two possible states per cell, and a cell's neighbors defined to be the adjacent cells on either side of it. A cell and its two neighbors form a neighborhood of 3 cells, so there are 2^3=8 possible patterns for a neighborhood. There are then 28=256 possible rules. These 256 CAs are generally referred to using Wolfram notation, a standard naming convention invented by Wolfram. The name of a CA is the decimal number which, in binary, gives the rule table, with the eight possible neighborhoods listed in reverse counting order. For example, below are tables defining the "rule

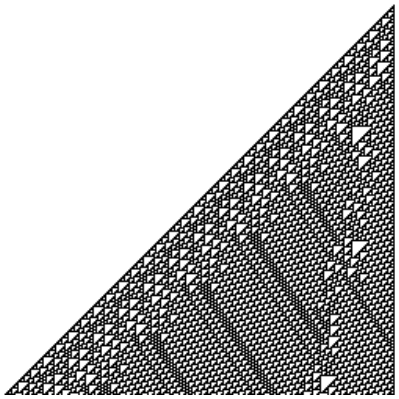
30 CA" and the "rule 110 CA" (in binary, 30 and 110 are written 11110 and 1101110, respectively) and graphical representations of them starting from a 1 in the center of each image [6, 7, 8]

A table completely defines a CA rule. For example, the rule 30 table says that if three adjacent cells in the CA currently have the pattern 100 (left cell is on, middle and right cells are off), then the middle cell will become 1 (on) on the next time step. The rule 110 CA says the opposite for that particular case.



Rule 30 cellular automaton

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	0	0	1	1	1	1	0



Rule 110 cellular automaton

current pattern	111	110	101	100	011	010	001	000
new state for center cell	0	1	1	0	1	1	1	0

Theoretically any rule can be used to generate random sequence of bits. In this research rule 30 has been used as the random key generate, where is proved by other papers and by the randomness testing shown below that it gives a good randomness. After applying the selected rule, and after number of steps (depends on the width of the array where the rule is applied), a rectangle will be appeared, any row or column in this rectangle may be used as a random key. One of the generated keys by applying rule 30 is tested by the formal randomness tests, and the result was as shown in table (1).

1- FREQUENCY TEST	Pass with value 1.054 must be <=3.84
2- ***RUN TEST***	Pass with value TO = 2.650 must be <= 5.702 Pass with value T1 =2.893 must be <= 7.531
3- POKER TEST	Pass with value 5.34 must be <=11.1
4- ***SERIAL TEST***	Pass value 0.833 with freedom degree "3 " must be <= 7.81
5- ***AUTO CORRELSTION TEST***	Shift No. 1>--> Pass value 0.000 Shift No. 2>--> Pass value 0.065 Shift No. 3>--> Pass value 0.000 Shift No. 4>--> Pass value 0.918 Shift No. 5>--> Pass value 0.215 Shift No. 6>--> Pass value 0.105 Shift No. 7>--> Pass value 0.642 Shift No. 8>--> Pass value 0.281 Shift No. 9>--> Pass value 0.500 Shift No. 10>--> Pass value 0.150 With freedom degree "1" must be <= 3.84

5. Improved 4-States operation

To increase the security and key space, that makes the encryption algorithms more robustness to the intruders, a new manipulation bits process has been added in [8] by using different truth table for manipulation bits process work on 4-states (0,1,2,3) , while the traditional binary process (XOR) work on (0, 1) bits only. The symbol # has been used to refer to the operator that execute this process use truth tables that shown in figure (2) [8].

The new operation needs 3 inputs, the first one specify the table number that should be used to calculate the result among the 4 tables, the other 2 inputs define the row and column number in the specified table where the cross point of them gives the result. An example of applying the operation is shown below:

Input 1: 0 1 3 0 1 2 2 3 1
Input 2: 3 2 2 1 0 1 2 1 1
Input 3: 1 0 0 2 1 3 2 1 2

Result : 1 2 0 0 1 2 2 3 2

#0	0	1	2	3
0	3	2	1	0
1	2	3	0	1
2	1	0	3	2
3	0	1	2	3

#1	0	1	2	3
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

#2	0	1	2	3
0	2	3	0	1
1	3	2	1	0
2	0	1	2	3
3	1	0	3	2

#3	0	1	2	3
0	1	0	3	2
1	0	1	2	3
2	3	2	1	0
3	2	3	0	1

Figure (2): The truth tables for the # operation

6. The proposed algorithm to modify Blowfish using 4-states

This research proposed a new improvement to the Blowfish algorithm. The proposed improvement makes use of the new operation defined in the previous section, operation (#) applied during each round in the original Blowfish algorithm, where another key is needed to apply this operation at both sides, this key may come in binary form and convert to a 4-states key, or it may already come in a 4-states as that can be done with quantum channel.
Consequently, two keys will be used in each round of the original Blowfish, the first key K1will be used with the xL and Pi to produce the next left part. The second key K2 will be used with F(xL) and xR to produce the right part. So the new structure make use of three keys in each round; that generated in one running of one dimensional Cellular Automata using rule 30; all the needed keys could be retrieved from any row or column of the triangle generated by this rule.

These three inputs to the # operation should be firstly converted from 32 bits to a 16 digits each may be one of four states (0, 1, 2, 3), i.e., each two bits converted to its equivalent decimal digits see figure(3).

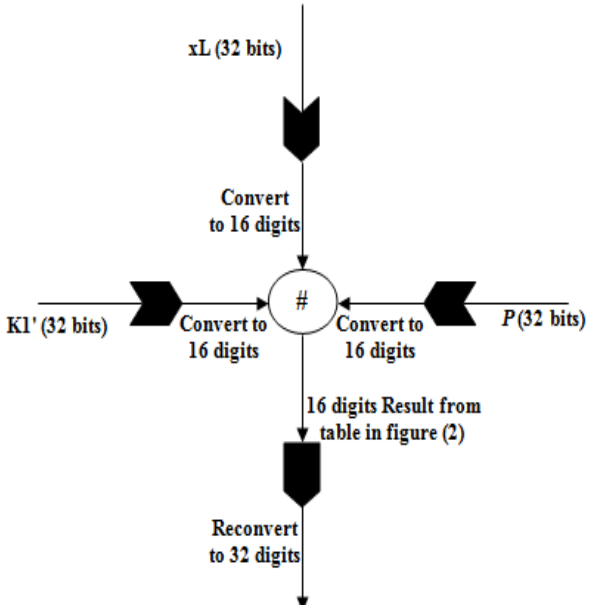


Figure (3): Inputs and Output of the # operation in DES algorithm

For example, the binary number:
10010111010100101010011111010001001
will be converted to the number:
2 1 1 3 1 1 0 2 2 2 1 3 2 2 0 2 1
Then the # operation will be applied to generate a new 16 digits that should be reconverted to 32 bits , see Figure (4). Full details of the proposed improved Blowfish are given in Algorithm (2) [8].

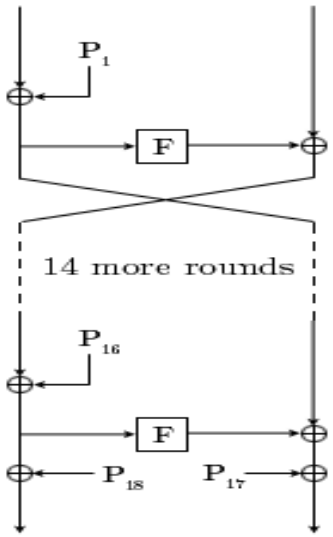


Figure (4): The new structure of each round in the improved Blowfish Algorithm

7. Implementation

The following example shows how the encryption and decryption operations results will be according to apply the # operation, so in any stage we can expect that the result of the previous left part of the data could be the binary number:

xL	**=**	**00101010010101011101010010111101110 0**
and the value of Pi-1 which represent here the first key could be the binary number:		
Pi	**=**	**10010100011101010101001001110100 1101**
and the entered key, which represent the second key in the applied # operation, the binary number:		
K1	**=**	**11101010010101011010111010101001011 1**
Firstly, the three entered 32-bits binary numbers should by converted to a 4-states 16-digits numbers.		
Pi '	**=**	**2 1 1 0 1 3 1 1 1 1 0 2 1 3 1 0 3 1**
xL '	**=**	**0 2 2 2 1 1 1 1 3 1 1 0 2 3 3 1 3 0**
K1'	**=**	**3 2 2 2 1 1 1 1 2 2 3 2 2 2 2 1 1 3**
Then the # operation applied according to tables in figure (2) the result of encryption will be:		
NewxL	**=**	**3 1 1 0 0 2 0 0 0 2 0 0 1 2 0 1 0 0**
If we reverse the whole operation we will get the initial, which is the result of the decryption operation that equal to the original data:		
K1'	**=**	**3 2 2 2 1 1 1 1 2 2 3 2 2 2 2 1 1 3**
Pi'	**=**	**2 1 1 0 1 3 1 1 1 1 0 2 1 3 1 0 3 1**
NewxL	**=**	**3 1 1 0 0 2 0 0 0 2 0 0 1 2 0 1 0 0**
xL	**=**	**0 2 2 2 1 1 1 1 3 1 1 0 2 3 3 1 3 0**

8. Conclusions

Blowfish is now considered to be insecure for many applications. So it becomes very important to augment this algorithm by adding new levels of security to make it applicable and can be depending on in any common communication channel. Adding additional key and replacing the old XOR by a new operation as proposed by this paper to give more robustness to Blowfish algorithm and make it stronger against any kind of intruding. One dimension Cellular Automata was used in this research for multiple random keys generation, where different random keys can be generated in just one applying of any CA rules, quality of PNSs highly depends on the applied CA rules, so rule 30 has been used which give more randomness. The ciphering process stills simple and can be implemented by hardware in this new proposed improvement, as well as the time complexity of the new algorithm stays the same since only one operation is replaced by another operation, and the conversion operations is very simple and straightforward.

Acknowledgments

The authors would like to thank the Al-Zaytoonah Private University of Jordan in Amman, Jordan for Supporting this publication.

References

[1] Alan G. Konheim, "COMPUTER SECURITY AND CRYPTOGRAPHY ",2007 , by John Wiley & Sons, Inc.

[2] Alfred J.M., Paul V. C. and Scott A. V., "Handbook of Applied Cryptography", Fifth Addition, 2001.

[3] Bruce Schneier, "Applied Cryptography, Second Edition: Protocols, Algorthms, and Source Code in C", 1996, Wiley Computer Publishing, John Wiley & Sons, Inc.

[4] B. Schneier, Applied Cryptography, John Wiley & Sons, New York, 1994.

[5] B. Schneier, Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish) Fast Software Encryption, Cambridge Security Workshop Proceedings (December 1993), Springer-Verlag, 1994, pp. 191-204.

[6] Coppersmith, Don. (1994). "The data encryption standard (DES) and its strength against attacks". IBM Journal of Research and Development, 38(3), 243–250.

[7] National Institute of Standards and Technology, (1979). “FIPS-46: Data Encryption Standard (DES).” Revised as FIPS 46-1:1988, FIPS 46-2:1993, FIPS 46-3:1999, available at http://csrc.nist7aznml;'.gov/publications/fips/fips46-3/fips46-3.pdf

[8] Hala Bahjat AbdulWahab1 , Abdul Monem S. Rahma, 'Proposed New Quantum Cryptography System Using Quantum Description techniques for Generated Curves", The 2009 International conference on security and management, SAM2009, July 13-16 2009, Las Vegas, USA, SAM 2009.

[9] Henk C.A. van Tilborg, Eindhoven, "ENCYCLOPEDIA OF CRYPTOGRAPHY AND SECURITY", 2005, Springer Science+Business Media, Inc.

Afaf M. Ali Al-Neaimi received the B.S. degree, in Mathematics from Al-Mostanseria University, Baghdad-Iraq, in 1978, and the M.S. in computer science from Iraqi Commission for Computers & Information, Baghdad-Iraq, in 1985, and Ph.D. degrees in software engineering from University of Technology, Baghdad-Iraq, in 2005. She is now an assistant professor at software engineering Dept., Faculty of Science and Information Technology in Al-Zaytoonah Private University of Jordan, Amman, Jordan. Her current research interests are in the area of software Engineering, ontology Engineering, Intelligent Systems, and computer security.

Rehab F. Hassan received the B.S. M.S. and Ph.D. degrees, in computer science from University of Technology, Baghdad-Iraq, in 1989, 1995 and 2005; respectively . She is now an Assist. Prof. in Computer Science Department University of Technology, her current research interests are in the area of systems software, data security.